Metamorphosis

Into a New Era

Vol 1, Nbr 7 | For the CoCo/OS9/OSK Communities | January 1994

Complete OS-9 File System Tutorial

Part VI of The Art of Programming

Memory Testing – Part Deux

And Much More!

January 1994, Vol 1, Nbr 7

Contents

Metamorphosis
Published monthly by
The Dirt Cheap Computer Stuff Company
1368 Old Highway 50 East
Union, Missouri 63084

Publisher
Mark D. Griffith
Editor
Barbara Ann Griffith
Contributing Writers
Mark Griffith
Mike Guzzi
Shawn Marolf
Boisy Pitre

Subscriptions
Metamorphosis is published 12 times a year. Subscription rates are $24 per year for the continental United States. Canadian rates are $32 (US) per year. Overseas rates vary. Please write for details.

How To Contact Us
Mail can be sent to the address shown above. Electronic mail can be sent via the Internet to:

76070.41@compuserve.com
or
MARKGRIFFITH@delphi.com

Telephone calls can be made to (314) 583-1168. Please leave a message and your call will be returned.

This publication is composed, formatted, and master pages created entirely on machines running the OS-9 operating system.

Features

Columns

Departments

On the Front Cover

There is no picture this month, to make room for the OS-9 3.0 announcement.

From The Editor's Desk

Ah, the month of January. The month that ends the Christmas season of cookies, cakes, hams, turkeys, gravy, presents, and endless shopping. The month that beckons in the new year and a time to start over, to make things right. Ah, yes, January. The month when the glitter and magic of the Christmas season are replaced with the cold, bitter facts of reality.

Let's start with New Year's Day. My grandmother has a family reunion every year on January 1st. This, as you might guess, is when everyone in the family gets together to celebrate another year being over and a new one beginning. This is when everyone tells you how much weight you have gained or how much more gray hair you have this year than last. This is when all those relatives that don't get along will get together and have it out so the rest of the family will have something to talk about over the next year.

After New Year's Day we must contend with the Christmas tree. Once beautiful and a delight to look at, it now has somehow turned from a work of art to just plain work. So, the tinsel, lights, and ornaments come off, along with about 50 million dead needles. I ask you now, is there a vacuum cleaner on the face of this earth that will pick up those blasted dead pine needles? I have yet to find one if there is. Oh sure, we give it our best shot, figuring if we only pick up a few needles at a time they won't clog up the mechanism somewhere along the line. But, ten minutes later, we're beating and banging the vacuum and shoving the broom handle down its throat. After this, we are forced to get down on our hands and knees to remove the dried needles from the carpet, our fingers, our feet, and yes, even our knees. I love this part so much.

So, now the Christmas decorations are put away, the tree is out of the house, and the dried pine needles are all picked up, except of course for those choice few which will be found by your bare feet over the next couple of months.

Next come the bills. You know, the little envelopes stuffed in your mailbox for things you bought to put under the tree. How could that little pile of presents under the tree generate so many envelopes? Ah, yes. Now you are beginning to see what I mean by reality.

And what about those New Year's resolutions? We spend time evaluating our many faults and resolve to change just one thing toward the good. I don't know about you, but come the beginning of February all of my resolutions have somehow been left behind in the month of January. My diet has failed, my exercise plan was just too exhausting to keep up, all those birthday cards I vowed to send early were forgotten entirely. So we come to the end of January feeling like a failure.

All of the above make for a pretty lousy month in general, but throw in the freezing, and I do mean freezing, temperatures we have been having this month and it's enough to make you want to pull your hair out.

My grandmother told me in the fall that the wooly worm said it was going to be a bad winter. I looked at the wooly worms crawling everywhere and wondered what she saw that I couldn't. They looked the same to me as they had in previous years. But, like it or not, Granny and the wooly worms were right.

We in Missouri have had below zero weather with wind chills down to -40 degrees. We have had to deal with burst pipes, faucets and drains that are frozen shut, and heaters running to the max to get the temperature in the house to a toasty 55 degrees. It's times like these I'm thankful we have the dogs to keep us warm. The term "it's a three dog night" would definitely apply in this case.

Ah, yes. January is the month that ends the Christmas season and beacons in yet another year of cold, bitter reality. Hold on to your checkbooks guys, April 15th isn't really that far away!

New Columnist

Beginning next month, in the February issue, a new columnist will join the *Metamorphosis* team. Mr. Ed Gresick will begin writing on the G-Windows graphical user interface for OS-9/68000. Ed, owner of **delmar corp.** is a well known OS-9 personality and friend to many. Ed also keeps his hand in on things by being a Director-At-Large for the **OS-9 Users Group.** Mr. Gresick will be a wonderful addition to our publication. Please join us in welcoming him.

Late Issues

We apologize for getting so behind on the issues. We had wanted to be able to mail each month's issue at the beginning of the month, but we got behind back in September and have never been able to catch up. We've decided to skip the December issue and get the January issue out ASAP. We'll extend everyone's subscribtion for another month to make sure you get your full 12 issue. The February issue will be out within three weeks and then, hopefully, we'll be back on track. We appreciate your patience and your support.

Keep warm everyone and we'll see you next month!

Mail Call!

Subscriptions Keep On Coming!

Dear Mr. Griffith:

Greetings. I would like to thank you for the recent copies of the *Metamorphosis* magazine. I am impressed with the format and the content of this young publication. It shows promise of growing into a long lasting periodical for our ever shrinking group. I considered producing something of the sort myself and decided it was one more project I would not be able to keep up to.

Which brings me to the purpose of this letter, speaking of keeping up to a project. I am including a copy of our NitrOS9 package. I would like it very much if you could have this reviewed and published in a future issue of your magazine. I liked the results of the article published comparing NitrOS9 to PowerBoost. Unfortunately, these results are inaccurate in terms of our latest version. Although our product appears to have the edge in this 'round,' the current patches actually do much better.

We ran the same tests Alan did in his article:

Test Screen	Stock	PowerBoost	v1.00/v1.07	v1.15
80 X 24 Text	41	36	28	20
80 X 24 2 color	96	75	52	37
80 X 24 16 color	258	205	169	127

I do plan on advertising in your magazine in the near future. My current financial situation has not allowed me to do so up to this point. I look forward to seeing my first ad in print in your magazine. Thank you for your time, and keep up the good work!

Wes Gale

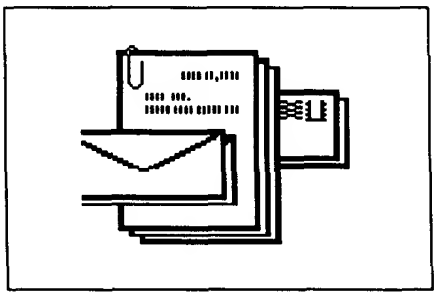
Thanks for your update Wes. A complete review of the latest version of NitOS-9 will be in the very near future. We hope you will like it.—Barbara

Dear Barbara:

Enclosed is my check for $24.00 for the next 12 issues of Metamorphosis. Congratulations with the magazine–keep up the good work.

Don W. Guido

Thanks Don! And thanks to all our readers for supporting us!—Barbara



Editorial Rebuttal

Dear Mark and Barbara:

I was particularly intrigued by the editorial on "Duplicate Articles" in different magazines and must point out a thing or two from the writers point of view. I have in the past submitted "by-lined" articles to the Port Orchard OS9 Newsletter and usually send these articles through the OCN for those that cannot afford these publications as I have not been able to for so many years. Also, in the event that I submit the same article to two different publications it is with (and only with) the understanding that there is no guarantee that my article will be published.

Since my submissions are free and intended to be free regardless of the availability of reimbursement funds, I really see no breach of ethics as was mentioned in your editorial. However, I would agree with your comments if the articles were submitted for the purpose of gaining compensation. I also agree with your comments since a work of Marks was submitted without his permission.

I feel that it is unfair to ask of us not to submit our articles to more than one publication when you yourselves cannot (or will not) guarantee us that our submissions will be published in the issue month in which they were received (by issue month I mean that if an issue mails on 10/01 and you receive an article from me on 10/02, then it would be published in the 11/01 mailing, on the other hand if you receive an article from me on 09/20, it would then appear in the 10/01 mailing).

Anyway, I do like your magazine and where you want to take it. Just don't fully agree with the above mentioned editorial. Onwards, and upwards.

Dave Gantz

You should examine a copy of **Writer's Market,** *a book written for those that would like to get their name in print. This book explains the rights of the writer and the publisher in detail. Since you obviously enjoy writing and submitting your articles, I suggest you might want to purchase a copy of your own. I find it quite useful myself. As I mentioned before, we cannot afford to purchase articles at this time. I understand completely what you are saying, Dave, but*

please tell me what you think we should do to avoid this sort of thing happening in the future. Should we hold every article submitted for a couple of months to be sure that articles sent to other magazines do not print during the same month we use the article? I do not think this is the answer. Perhaps we should only print those articles that we have a written contract for. I am not sure this would work either, since that would eliminate a lot of articles that we receive. Most of our writers are glad to have their article put into print and read by our subscribers. You, apparently, are heavily into writing and submitting articles so this is not the case for you. If you (or any of our other readers) could shed some additional light on this subject, we would be glad to hear your suggestions. We are still learning to walk, so to speak, and still need a hand now and then.—Barbara

News Clips

More Machine Rumors

My ear is still to the ground trying to pick up more information on the new OS-9 computer I mentioned last month. I learned that the project is a go at this time with the possibility of a pre-production board being available for show at the Chicago CoCoFest in May. I have also learned that the builders are looking into more advanced features like Ethernet support. Stay tuned for more as it comes in!

Article Submissions

As the new year begins, I find we are running out of articles to print. I have contacted a number of people and asked them to write an article or two in areas they have some expertise. So far only a couple have responded.

We really need the help of the readers. If you have a desire to write about something, even if you can't write a grocery list well, please contact us. We can help you make your article into something you'll be proud of. If you know of anyone else involved in the OS-9 world that likes to write, please have them contact us too. As we progress, we'll be able to start paying for articles. But we need to get to that point first.

This short list of possible article topics may help stimulate your brains. If you have any other suggestions, please let us know.

- Graphics files formats and conversions
- Internetworking, such as StG net, FIDO, etc
- Networking protocols like TCP/IP, NFS, and Microware's NFM
- Using termcaps
- TeX, Lout and other document processing packages
- OS-9 system management
- GUI systems and graphics processing (Kwindows and Gwindows)
- Porting UNIX applications to OS-9
- Backup. Which media is faster? Possibly include some C source for a backup scheme
- Article on X-10 home control modules

Mark Griffith

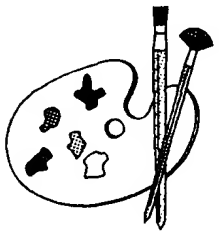
The Art of Programming

Part VI

Program Organization

by Shaun Marolf

Shaun Marolf, 30, is a member of the US Navy stationed at Naval Air Station San Diego where he is currently working on his degree in Computer Sciences. He first learned programming in 1979 on an Apple II Plus and is also schooled in electronics and digital circuits. Shaun runs the "Eight Bit Heaven" BBS (619-447-2111) and owns several computers, including an original grey case CoCo 1.



Modular programming requires great care and organization. In order to write an effective and efficient program, the author must be aware of what the program is doing at all times.

In Part One we covered the need to note all information concerning the functions and applications of the program. We also looked at how to set up the operator interfacing. We covered the gives and takes needed in setting up the program and how to balance them. Now it is time to start putting it all together.

First, let us look at the essence of the modular program. It is nothing more than a series of top to bottom programs which are linked together through a master or core module. Each module performs a specific take within the whole of the program. The core is also the main interface between the user and the operating system.

The first thing that needs to be done when designing the modular program is to separate the separate functions and operations within it. This will give us the building blocks needed to write the program. Make every attempt to keep the modules as isolated as possible. By isolation I mean know exactly what is going in the module and also know exactly what should come out. By doing this, the module becomes much easier to make changes to without affecting others. However, not all modules will be easily isolated, and some will be downright impossible. The tighter you keep them the better off you will be.

The core module must be started first, but you will need to start with a crude version. Since the core is the master, I find it easier to work the I/O of other modules into the core as I put them together into the master itself. I then add all the routines into the core as I test each module interface. This may not necessarily be the best method for you, but no matter how you do it there will always be needed changes to the core as you add the separate modules. The core should be your largest and most difficult module, since it performs so many things. Remember, this is the heart of your program.

In some cases the module may have a need to interface with itself. If this does happen within your program, it becomes vital to watch the data during the initial setup of the program to ensure that the data is not irrevocably altered.

The best advice I can give is to treat each module as a separate program, since in essence that is what they are. Just remember that a module must interface and exchange data with other modules. Also, keep in mind that you the programmer are now responsible for the logic and data flow. The program will not automatically fall into place so it is up to you to make sure that all processes fall into a logical order. If you don't, you can expect things to go wrong.

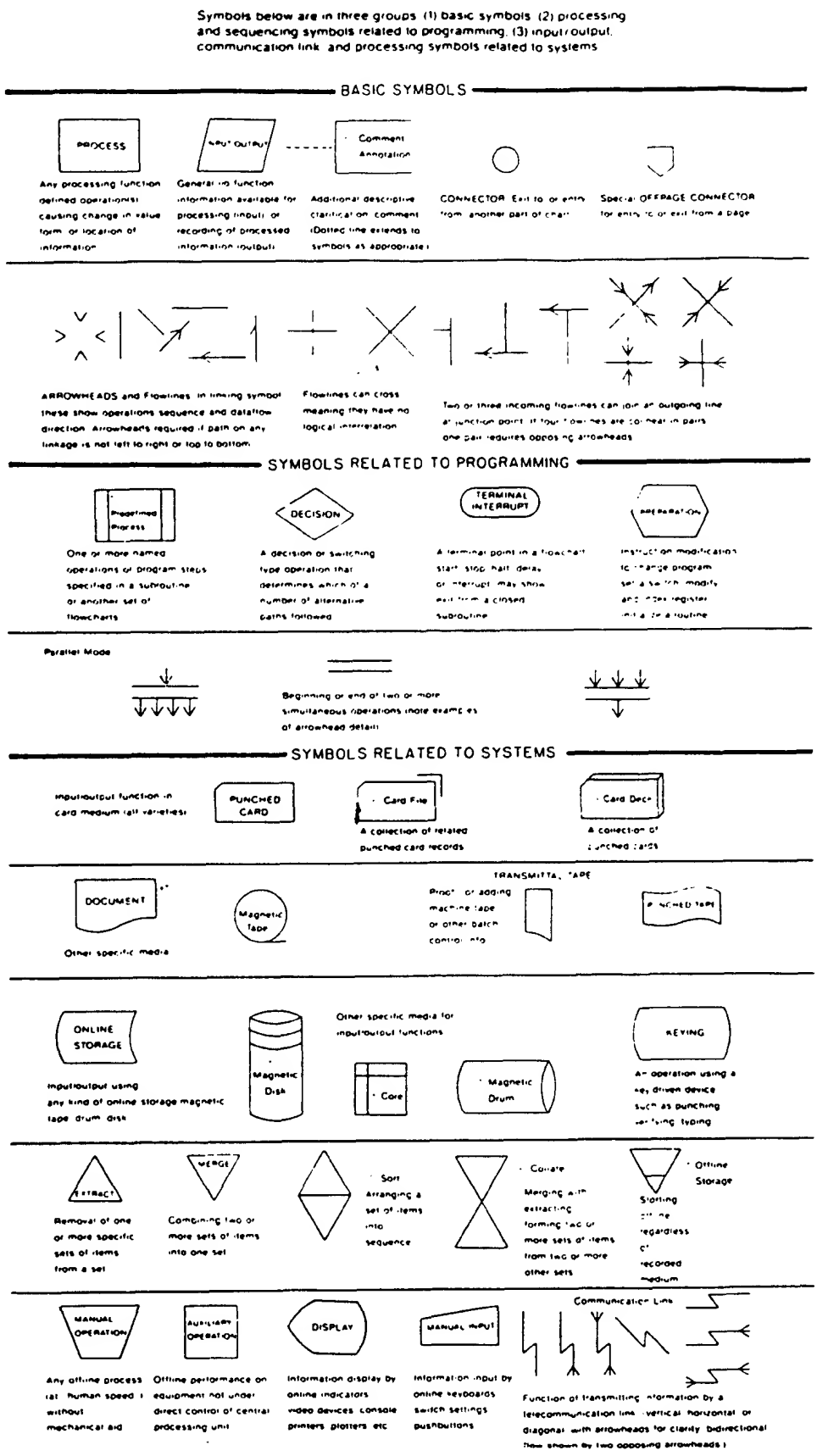
Returning to the beginning installments, it is important to keep notes and redo the flowcharts when changes are made. Wordstar (then Micropro) Incorporated learned this lesson well. After years of sitting on Wordstar 3.0, where no effort was made to make any improvements, they found that Wordperfect was slowly overtaking them in the market. When the decision was made to upgrade the program, it was discovered that no notes were taken on the program, and that all the original programmers had left to form their own company. By the time everything got straightened out, Wordperfect had become the dominate seller, and Wordstar never regained its status as the word processor of choice. All because notes were not properly kept on one program.

Even if you are the sole programmer, notes will help you keep track of things when you decide to make revisions and updates. I am including two full-page forms with this installment to help you

keep your program notes better organized. The first is a program worksheet which is designed for initial conception ideas and designing of the program functions. It is also good for note keeping while flowcharting and coding your program. The second one is a flowchart worksheet and is fairly self-explanatory. I highly encourage you to photocopy these forms and make full use of them.

I am also including a chart of the most commonly used flowchart symbols with a symbol variations and usage chart to help further expand the flowchart's readability.

We will start wrapping things up and cover the phases of programming in the next installment.



From The Jargon File

Conway's Law *prov.* - The rule that the organization of the software and the organization of the software team will be congruent; originally stated as "If you have four groups working on a compiler, you'll get a 4-pass compiler."

This was originally promulgated by Melvin Conway, an early proto-hacker who wrote an assembler for the Burroughs 220 called SAVE. The name 'SAVE' didn't stand for anything; it was just that you lost fewer card decks and listings because they all had SAVE written on them.

OS-9 Version 3.0

The Best Has Gotten Better

by Boisy Pitre

Boisy Pitre, 22, is a software engineer at **Microware Systems Corporation** *and an ex-president of the OS-9 Users Group, Inc. Boisy can be contacted at PO Box 523, Waukee, IA 50263 and as boisy@delphi.com.*

A major software release is guaranteed to usher in a variety of changes and features – such is the case with the new OS-9 3.0 for 680X0-based systems. In spite of this, the software is solid, the enhancements are superb, and compatability superior, with customization and configurability being the hallmarks of this new release.

The evaluation system used for this overview consisted of a 68030/20MHz processor board with Quantum 240MB SCSI hard drive, TEAC tape drive and four serial ports. The time it took for me to complete the process of installing the OS-9 3.0 Development Pak (from software installation to boot-up) was approximately four hours.

At the time of this writing, I have been using OS-9 3.0 on the evaluation system for five weeks; relentless software development, testing and 24-hour operation have taxed the reliability and durability of the software. To date, I have encountered no problems with the software.

Installation

The OS-9 3.0 Developer's Pak came on nine 3.5 inch universal-formatted floppy disks. One change noticed immediately after installation is Microware's implementatin of a new directory structure: MWOS. This tree is located in the root directory and branches out to accommodate all of Microware's products in a logical and organized directory structure. Getting used to this new structure may take some time, but the benefits are soon realized. Underneath the MWOS directory are other directories that encompass OS-9 and OS-9000 supported processors, allowing for code sharing between the two operating systems.

The Kernel

Choosing the kernel is the first step in configuring the system for OS-9 3.0. Microware has created separate kernels for the following processors: 68000, 68010, 68020, 68030, 68040, 68070, CPU32 and 68349. Optimizing the kernel for a particular processor brings obvious speed gains to the system, and that's just one of the many facets of optimization that OS-9 3.0 has gone through.

Atomic vs. Development

In addition to a CPU-specific kernel, Microware offers two types of kernels: development and atomic. The development kernel gives the user the full range of OS-9 services, while the atomic kernel retains the core characteristics of OS-9 yet is specifically tailored to embedded systems. User-state debugging calls (F$DFork, F$DExec, F$DExit) are not supported by the atomic kernel. There is also no support for multi-user protection mechanisms (e. g. module permission checks).

The idea behind the development/atomic kernel method is that the embedded programmer uses the development kernel while developing the application. Once fully debugged and ready for production, the smaller and faster atomic kernel then replaces the robust kernel. In practice, the personal OS-9 user will want to use the development kernel.

Memory Management: Standard vs. Buddy

Once you've decided which kernel to use, you must decide on the type of memory management scheme you wish to implement. Two types are provided for both development and atomic kernels: the buddy allocator and the standard allocator.

The standard memory allocator is the same as that used by earlier versions of OS-9. That is, it can allocate memory down to a resolution of 16 bytes. This type of allocation is usually slower, but efficiently allocates and frees system memory. This is the allocator algorith of choice for personal OS-9 users.

The buddy allocator is a faster memory allocation scheme, but is less efficient in terms of memory fragmentation. It allocates memory by rounding up the request to the next binary power. For example, a request of 1032 bytes is rounded up to 2048 bytes. This allocation method is more suitable for embedded applications where determinism is of great importance.

It is also worthy to note that both memory schemes now support two new color types: NVRAM (non-volatile RAM) and SHARED (shared RAM) memory.

IOMan Returns

IOMan is back as a separate module. Once part of the kernel, Microware engineers decided to make it a stand-alone module. Doing so gives embedded developers more available ROM space in case their application doesn't need an I/O subsystem or in the

event that a customized I/O manager is used. Atomic- and development-specific IOMan modules exist for either kernel.

System-State Time-Slicing

OS-9 can now perform time-slicing while in system-state, making CPU usage fairer and system determinism more predictable. This feature is optional and can be turned on or off either system wide in the Init module, or on a per-process basis in the process descriptor. Release notes provided with OS-9 3.0 gives information for those programmers who wish to take advantage of the system-state time-slicing feature in their system-state programs. The release notes also encourage developers to relink system-state code to ensure that structure and definition changes are properly resolved.

Backward Compatibility

Surprisingly, there is very little compatibility conflict between OS-9 2.4 and OS-9 3.0. There are some changes that may need to be made, however. For 68020/030/040 CPUs, the kernel uses the master stack pointer (MSP) bit found in the status register. Since pre-3.0 drivers usually write over this bit when masking and unmasking interrupts, they must be modified to preserve the MSP bit when masking the interrupt level.

Because I use Carl Kreider's excellent serial and disk drivers (which do NOT preserve the MSP bit), it was necessary for me to alter the source code and remake the drivers. The code excerpt below shows the wrong way to build an SR image for interrupt masking.

```
moveq.l  #0,d0                clear register
move.b   PD_M$IRQLvl(a1),d0   get irq level
lsl.w    #8,d0                move level to correct SR position
bset.l   #SupvrBit+8,d0       force system-state
move.w   d0,MaskIRQ(a2)       save SR image for later

later...

subq.l   #4,a7                make SR save space
move.w   s4,0(a7)             save current IRQ level
move.w   MaskIRQ(a2),sr       mask interrupts

later...

move.w   0(a7),sr             restore SR
addq.l   #4,a7
rts
```

The code below properly preserves the state of the MSP bit when building the SR image.

```
moveq.l  #0,d0                clear register
moveq.l  #0,d1                clear register
move.w   sr,d1                copy current SR
andi.w   #IntEnab,d1          clear interrupt flags
move.b   M$IRQLvl(a1),d0      get device irq level
lsl.w    #8,d0                move level to correct SR position
or.w     d0,d1                form sr image
move.w   d1,MaskIRQ(a2)       save it for later
```

These changes are not needed for 68000/68010/68070 processors since they do not use the MSP bit; however, it is a good idea for driver developers to practice this coding method no matter what processor is used. Boot ROMs from OS-9 2.4 may also be susceptible to change, since pre-3.0 versions of rombug did not properly support the MSP register bit. Pre-2.4 boot ROMs will not work with OS-9 3.0.

Other Changes/Enhancements

- Semaphores have been added to the kernel.
- The process descriptor has been changed from a fixed 2K size to a dynamic structure.
- Sleep(0) and sleep(1) are much faster.
- Process scheduling has been improved.
- Pipe transfers are much faster due to pipeman performing direct I/O calls to the buffer when possible.
- System-state alarms are much faster.
- Bitmap services are much faster.
- Process and path table searching for free entries are much faster.
- Data modules are no longer created with a valid CRC. The CRC is initially to zero.
- Interrupt response times have been improved for F$IRQ installed devices, as well as the implementation of a fast interrupt scheme (F$FIRQ). The F$FIRQ routine, featured in the atomic kernel, saves fewer registers and cannot anticipate exceptions, resulting in very fast response time
- The Init module has been changed to reflect new features and remove obsolete ones
- The csl and fpu trap handlers are provided for OS-9 utilities, since relevant parts of the package have been recompiled with Ultra C

Conclusion

Due to the number of packaging options available, I have deliberately omitted pricing. Feel free to contact your Microware sales representative for pricing and packaging information on OS-9 3.0 and other Microware products.

Finally, for more information on OS-9 3.0, read Dr. Peter Dibble's in-depth article titled *Techniques For Optimizing A Real Time Operating System* in the October 1993 issue of **VMEbus Systems,** pp. 13.

OS-9 File Structure

Part II

by Mike Guzzi

Mike Guzzi, 25, has a BS in Electrical Engineering, and is currently a technician at Specialty Records Corp/WEA Manufacturing INC. Mike wrote the APBBS BBS system that was sold by Second City Software. He has made several contributions to the public domain including Mbackup which allows a CoCo user to use more than 64k for copying a disk legitimately, under OS-9. Mike has owned a CoCo since 1984. He can be reached at **mike_guzzi@delphi.com.**

The structure of an OS-9 disk is very different than a disk for Disk Extended Color Basic. The biggest difference is you can have more than one directory on the same disk. Let's say you have three disk drives attached to your system. Think of it as a file cabinet with three drawers. The drawers would be labeled as /D0, /D1, and /D2. These would represent each disk drive on your system. When you open a drawer, inside will be folders; these would be the names of directories on an OS-9 disk. Inside these folders you will see papers containing data. These would be called files in OS-9. Let's make up an imaginary disk for running a small business. We will use the first drawer, and inside it we will need several folders. The first will need to be for payroll so you make a folder and label it "PAYROLL." A second folder would be needed for accounts labeled "ACCOUNTS." A third could be for bills so let's call it "BILLS." So how does this translate to a disk? You have a disk in the first disk drive. The drive is called "/D0." The folders will be directories on the disk called "PAYROLL" "ACCOUNTS" and "BILLS." Inside each directory are the files with data. As you can see, this provides a way to organize information on a disk to find it later. Under Disk Basic, you only had the main directory and that was it; all your files HAD to go there. It would be like a file cabinet with only one folder.

Let's take a look at your OS-9 disks. Boot up OS-9 and type the following:

```
OS-9:dir /d0
```

You will see the following listing.

```
directory of /d0   11:38:12
OS-9Boot     CMDS         SYS              startup
window.t38s  window.t80s  window.glr4
```

The two names, CMDS and SYS are directories, while the other names are files. The way to find this out is to ask OS-9 for a complete directory listing. To do this type the following:

```
dir /d0 e
```

You will see a listing like this:

```
directory of /d0   11:41:27
Owner Last modified attributes sector bytecount name
----- ------------- ---------- ------ --------- ----------
  0  87/02/16 1648  ----r-wr        A       69E3 OS-9Boot
  0  92/05/19 1846  d-ewrewr       75        620 CMDS
  0  87/02/16 1654  d-ewrewr       7E        140 SYS
  0  86/08/13 1447  ----r-wr      237         C0 startup
  0  86/10/22 1606  ----r-wr      239        117 window.t38s
  0  86/10/22 1605  ----r-wr      23C        168 window.t80s
  0  86/10/22 1628  ----r-wr      23F        2B0 window.glr4
```

The "e" part of what you typed tells the dir command to give you the entire information. Look at the column marked "attributes." You will see that the two names CMDS and SYS have a "d" in the first column. This indicates that these files are in fact directories. What is inside these directories? Well let's take a look. Type the following:

```
OS-9:dir /d0/cmds
```

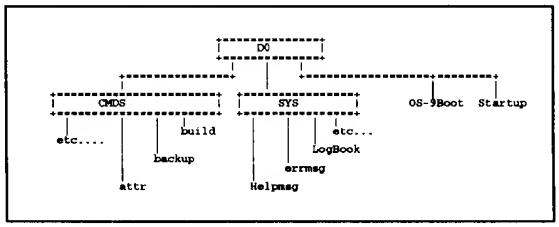
You will see a listing like this:

```
directory of /d0/cmds   11:47:19
attr      backup    build     cmp       cobbler
copy      date      dcheck    deiniz    del
deldir    dir       display   dsave     echo
edit      error     format    free      grfdrv
help      ident     iniz      link      list
load      makdir    mdir      merge     mfree
modpatch  montype   OS-9gen    procs     pwd
pxd       rename    setime    shell     tmode
tuneport  wcreate   unlink    xmode
```

You can do the same for the SYS directory by typing "dir /d0/sys" you will get a listing like this:

```
directory of /d0/sys     11:49:02
errmsg      helpmsg     LogBook    stdfonts
stdpats_4   stdpats_16  stdptrs    stdpats_2
```

So how do you access these files? Let's draw a map of this disk which shows how the files are stored on the disk.



This example only represents a small part of the disk. There are a lot more files in each directory, but I want to keep this example simple. So now let's say you want to access the file called backup. How do you tell OS-9 to find the file? You give it what is called a pathlist which is a name that describes the path OS-9 must take to find the file, much like a path on a map. Each directory is separated by a slash '/'. the name /D0 is the device where OS-9 will start from, in this case, the first disk drive. The pathlist follows:

```
/D0/CMDS/backup
 |    |      |
 |    |      +== name of the file.
 |    |
 |    +=== name of the directory
 |
 |
 +== Drive name: /D0
```

To get to the file called "LogBook," we take a different path, like this:

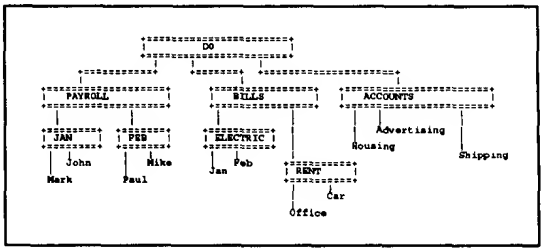
```
/D0/SYS/LogBook
```

To get to the file called startup we take this path:

```
/D0/startup
```

You can think of directories as forks in a road. You can go to multiple destinations from a fork in a road. When you give OS-9 a pathlist, you are telling OS-9 which roads to follow to get to the destination. You can make as many directories as you want. You will also notice that directory names are in all capital letters and files aren't. This is not mandatory, but it helps you to spot what is a file and what is a directory much easier.

Let's take the imaginary example from above and let's make up an OS-9 disk to organize our office system. We will start with a blank disk in the first disk drive called "/D0." Inside it we will want several folders for various things such as bills, payroll, accounts, and so forth. Here is an example map:



Type the path it needs to follow to get to the file, like this:

```
/D0/PAYROLL/FEB/Mike
```

What if I need the advertising information? It can be found like this:

```
/D0/ACCOUNTS/Advertising
```

Being able to create directories like this, you can see how neat and organized this disk is. Each file is in a clear and concise area on the disk according to what information the file holds. The disks that came with OS-9 are organized in a similar way. One point that might be brought up is that to get to these files would require a lot of typing. A pathlist could be rather long like this:

```
/D0/ADVERTISING/JANUARY/AGENTS/John.Doe
```

That is a lot of typing to get to the file john.doe however OS-9 provides a way to access a directory without all the typing. Taking the example from above, if you needed to access January's electric bill you would have to type the following:

```
/D0/BILLS/ELECTRIC/jan
```

However, OS-9 has what is called "current working directory." which is a way to shorten the pathlist needed to access your files. the command to use is called "chd" this stands for "Change Working Directory." If you needed to access files concerning electric bills you can type the following:

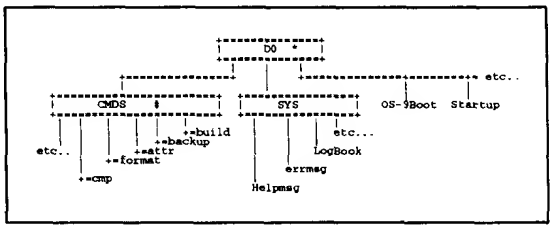
```
chd /d0/bills/electric
```

Once this is done the pathlist for the file Jan becomes this:

```
Jan
```

Basically what you did was move an imaginary pointer so OS-9 can find your files without needing the entire pathlist, and therefore, less typing.

Remember when I said above that OS-9 stores a lot of its information on disk? What this means is a lot of the commands you type are stored on disk and OS-9 must load it into memory before it can run it. OS-9 needs to be able to find these files also, called commands. These are files that contain programs that OS-9 runs to do a task. Under Disk Basic, all commands are in memory such as DSKINI0, DIR, LOAD, SAVE, and so forth. OS-9 stores most of its commands on the disk. There is a pointer that tells OS-9 how to find these programs just like a pointer to find the files. This is called the "current execution directory." It is a directory that contains files that OS-9 can execute as a program. This is probably the most confusing aspect of OS-9 and causes a lot of problems if its not understood. So let's look at your OS-9 system disk. Boot up OS-9 according to the instructions given in the book and I will show you how this works. When you have booted OS-9 it sets up defaults for the current working directory and the current execution directory. The current execution directory is "/D0/CMDS" and the current working directory is "/D0." Let's take a look at a map of your system disk:



I'll use a '*' to represent the "current working directory" and a '#' to represent the "current execution directory." This map shows what OS-9 set up as its defaults after booting the system. So in the directory called CMDS you will find programs that OS-9 can execute. Let's take a look, type this:

```
dir /d0/cmds
```

You will see the following.

```
directory of /d0/cmds   11:47:19
attr      backup    build     cmp       cobbler
copy      date      dcheck    deiniz    del
deldir    dir       display   dsave     echo
edit      error     format    free      grfdrv
help      ident     iniz      link      list
load      makdir    mdir      merge     mfree
modpatch  montype   OS-9gen    procs     pwd
pxd       rename    setime    shell     tmode
tuneport  wcreate   unlink    xmode
```

These are all programs that will perform a task. Let's say you wanted to see the contents of a file containing text. The list command will do that However if you type "list" you will get ERROR #215. Why? The list command needs to know what file to display. This is called a parameter. Let's look at an example OS-9 command line.

```
OS-9:list /d0/startup
  |   |       |
  |   |       += This parameter tells 'list' what to display
  |   |
  |   += This is the command to execute, in this case 'list'
  |
  += This is the prompt, you don't type this part
```

You can actually type this in. You'll see the following:

```
* Echo welcome message
echo * Welcome to OS-9 LEVEL 2 *
echo * on the Color Computer 3 *
* Lock shell and std utils into memory
link shell
* Start system time from keyboard
setime </1
date t
```

The file 'startup' contains these lines of text. However, remember that your current working directory is set to "/d0" when you boot up, therefore you can accomplish the same thing by typing:

```
list startup
```

Ok so far so good, but looking at the map above, let's say we want to see what's inside the file called LogBook. How can we list its contents? There are two ways to do it:

```
list /d0/sys/logbook
list sys/logbook
```

Both do exactly the same thing. Remember the working directory is set to "/d0" so we didn't need to type the "/d0" part. We can even shorten it further by first typing this:

```
chd /d0/sys
```

Or even simpler....

```
chd sys
```

Remember the '*' pointer? We just moved it to the box labeled 'SYS'. Now you can list the file by typing this:

```
list logbook
```

If you do type this, you will see the following:

```
Product: OS-9 Level II Operating System
Release Date: 02/16/87   Version: 02.00.01
```

OS-9 has an easy way to refer to its current working directory. You can use a dot '.' to refer to the current directory you are pointing to. For example, to display the current directory, you can type

```
dir .
```

or simply

```
dir
```

Dir will assume the '.' if you omit it. All commands that use a filename for a parameter can be referred to in this fashion. Let's take another example. Look at the following two commands.

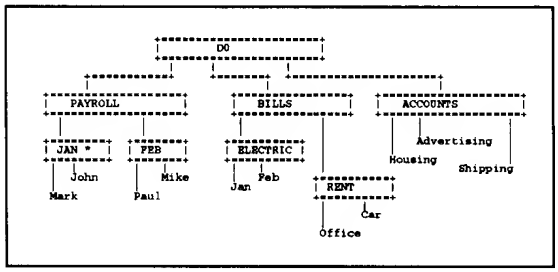
```
list logbook
list ./logbook
```

Both commands do the same thing. Since above you had done a "chd /d0/sys" The dot is the /d0/sys part, and you simply added the /logbook part. The first command is easier to type. OS-9 assumes the current working directory. How can you tell? If you do not have a slash '/' as the first character, you are referring to the current working directory. If you do use a slash the first name must be a drive name, such as /D0. This is referred to as an absolute pathlist. You are telling OS-9 the absolute path to find a file. This will become more clear later when you need to access files that are not on the current disk.

It is also possible to move around on the disk by using more dots. For example if you use two dots such as '..' you are telling OS-9 to backup one level. Let's look at a few examples. Say you did the following commands using the imaginary example from above.

```
chd /d0/PAYROLL/JAN
```

Looking at the map, this is where we are.



Notice where the '*' is. That's the current working directory. So if you needed to access files in that directory, commands like

```
list John
list mark
```

will work fine. What happens if you need to look at Paul or Mike? Well you could specify the absolute pathlist to get to these files such as.

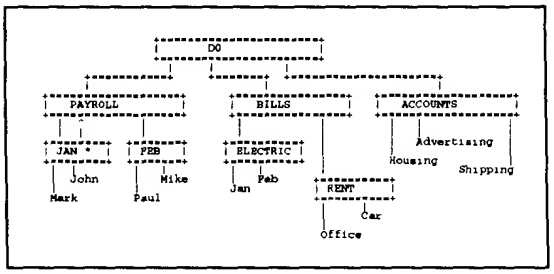
```
list /d0/payroll/feb/paul
list /d0/payroll/feb/mike
```

But, there are shortcuts. Most of the pathlist IS the current working directory. However, it's the last part (jan) that's incorrect. By using the anonymous names (dots) we can use a relative pathlist to find what we need. Ok so from our current position, how can we get to the path we need? If we use two dots '..' that will move us up one level to the PAYROLL directory. Then we need to move back down to the FEB directory, then

to the actual file we need. So how do we construct this? Look at the following example.

```
list ../feb/mike
```

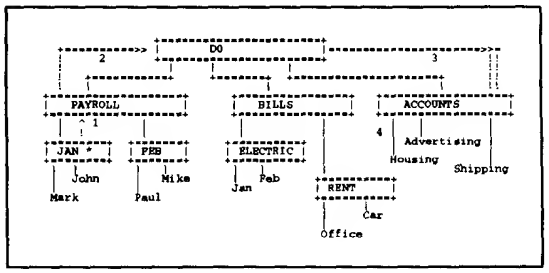
Let's look at a map, and see exactly what this pathlist does. You will notice that the current working directory DOES NOT CHANGE.



As you can see, we start with the current working directory. The two dots tell OS-9 to move up one level to the PAYROLL directory, from there we go down to the FEB directory, Finally, we get to the file Mike. We can extend this even further. What if we need to get to a file in the ACCOUNTS directory? By adding another dot, we can move up another level. here would be the correct command line to list a file in the ACCOUNTS directory.

```
list .../accounts/housing
```

Let's look at the map and see what OS-9 did.



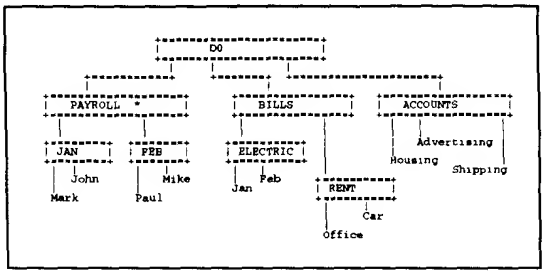
In the first and second part, the three dots moved us all the way back to the root directory (/d0) then the third part moved us down the ACCOUNTS directory and finally the last part, accessing the actual file we need.

The above shows how to access files when you need to move up levels to get to needed files, but what about moving strictly down levels? For example let's do this imaginary example. Looking at our above example, let's change our working directory to the next level above. From the above we can do it two ways. We can do it using the absolute pathlist, or we can simply specify it relative to where we are now. Here are the two ways to do it.

```
chd /d0/payroll
chd ..
```

Both commands will do exactly the same thing. If you're not sure why the second command works, study the section above.

Iur current working directory is now where we want it. Let's explore how we can access files down lower. This is what our map will look like now.



Notice the asterisk is now at the PAYROLL directory. Ok let's say from that point we want to look at files in both JAN and FED directories? Of course you could use the absolute pathlist and such, but how can we do it relative to where we are? Look at the following two examples:

```
list jan/john
list feb/paul
```

That would look similar to an absolute path but there is one difference. If you have to specify an absolute path, the first character is ALWAYS a slash. Another thing, if you do specify an absolute path, the first name in the path is ALWAYS a drive name, such as /D0 /H0 /D1 and so forth. Looking at the above commands, both JAN and FEB are directories below the current point where we are sitting.

This should give you an idea how to crawl around a disk directory, but since many use OS-9 with only floppy disks, you will need to access files that are not on the disk. This part tends to confuse people since each disk has a unique directory structure. If you change disks, you must do one of two things. Immediately change your working directory to the new disk, or you will have to specify an absolute path for every file you want to look at. This is important even if your current working directory is the root directory. (such as /d0 /d1 etc..) because there is a possibility the root directory is not in the same place. Most of the time it will work anyway, but it's a good habit to form. What you are doing is informing OS-9 that you have changed disks.

Next month we'll continue with our search through the OS-9 directory tree.

Hardware, From the Inside Out

Testing Memory, Part Deux

by Mark Griffith

Last month I talked about memory checking software. I also mentioned that the MM/1 was rather unique in the computer world in that you can install additional RAM in the system, but not use it. This allows you to use the expansion SIMM sockets as a sort of memory testing station.

The MM/1 uses a linear address space as do most modern computers. On a 1 Megabyte system, all the available RAM is addressable from $000000 to $0fffff. The 2 Megabytes of expansion RAM is then located from $100000 through $2fffff.

So, we would define our addresses as follows:

```
#define MEM_START   0x100000
#define MEM_END     0x2fffff
```

Next we deal with reading and writing directly to memory in C. This is not difficult, as long as the variables are correctly defined. When we're manipulating chip registers and reading memory locations, we usually only want to work with one byte at a time. Declaring a variable as type *char* can create problems if the value of that variable exceeds the upper and lower limits of -127 to +127. To make sure we don't get caught by this difficult to find bug, we declare our variables to be type *unsigned char*.

Since we have declared the memory start and end variables as type *integer* (by default), we'll need to cast them to an *unsigned char* when using them within our programs. This may seem to some like a waste since the program will run correctly without this, but these are good programming practices to develop. You'll prevent lots of headaches in the future. When dealing with hardware, it is especially important since an errant pointer can cause real havoc with the system—to the point of damaging files or even the hardware itself.

Now to the guts of our little program. All we need to do is to set the memory to some easily recognized character, wait a little bit, and then start checking it to see if anything has changed. To make it easier, I just used the `memset()` function to set the memory to a hex $48, or the 'H' character. Use `sleep()` to wait one minute, and then simply check the memory. If one or more bytes have changed, something is wrong.

Below is the entire program. Try it out. Next month, we'll go a little deeper. Until then.

```
/*  memtest1.c - Tests memory refresh */

#include <types.h>

#define MEM_START   0x100000            /* Start of expansion RAM */
#define MEM_END     0x2fffff            /* End of user RAM for a 3 Meg system */
/*#define MEM_END     0x7fffff          /* End of user RAM for a 9 Meg system */

main()
{
    u_char  *address;
    int     mem_size;
    address = (u_char *)MEM_START;
    mem_size = MEM_END - MEM_START;

    memset (address, 'H', mem_size);     /* set memory to all one value */
    sleep (60);                          /* wait for one minute */

    do {
        if (*address != 'H')
            printf (Bad  Memory at Address %Xn, address);
    } while (address++ != (u_char *)MEM_END);
}
```